LArSoft - Support #25606

Libtorch v1.6.0 high memory usage

03/09/2021 01:39 PM - Alex Wilkinson

Status:	Resolved	**Start date:**	03/09/2021
Priority:	Normal	**Due date:**	
Assignee:		**% Done:**	100%
Category:	External Packages	**Estimated time:**	0.00 hour
Target version:		**Spent time:**	0.00 hour
Experiment:	-	**Co-Assignees:**	

Description

Libtorch v1.6.0 from larsoft cvmfs uses 4 times more memory than a standalone build of libtorch v1.6.0 built by doing:

```
curl -o libtorch-shared-with-deps-1.6.0%2Bcpu.zip https://download.pytorch.org/libtorch/cpu/libtor
ch-shared-with-deps-1.6.0%2Bcpu.zip
unzip libtorch-shared-with-deps-1.6.0%2Bcpu.zip
```

Using the CMakeLists and test script provided, to reproduce first build with standalone libtorch:

```
mkdir build
cd build
cmake -DCMAKE_PREFIX_PATH=<path to standalone libtorch> ..
cmake --build . --config Release
```

Then to build with larsoft libtorch:

```
setup libtorch v1_6_0 -q e19
mkdir build_lar
cd build_lar
cmake -DCMAKE_PREFIX_PATH=/cvmfs/larsoft.opensciencegrid.org/products/libtorch/v1_6_0/Linux64bit+3
.10-2.17-e19 ..
cmake --build . --config Release
```

To test I have been using:

```
OMP_NUM_THREADS=1 /usr/bin/time -v ./test_tscript /dune/app/users/awilkins/infill_work_updated/une
tdense_induction_10e_020321.pt
```

The larsoft build reports "Maximum resident set size (kbytes): 9198012" and "Elapsed (wall clock) time (h:mm:ss or m:ss): 0:39.15" while the standalone build reports "Maximum resident set size (kbytes): 2340212" and "Elapsed (wall clock) time (h:mm:ss or m:ss): 0:28.81". Using v1_6_0a gives similar results. I don't expect to be needing ~2Gb networks but the discrepancy in the builds seems concerning. Thanks very much for any help.

History

#1 - 03/09/2021 01:42 PM - Alex Wilkinson

- File CMakeLists.txt added

- File test_tscript.cpp added

#2 - 03/15/2021 10:42 AM - Erica Snider

- Status changed from New to Feedback

Hi Alex!
We're concerned about the manner in which your downloaded version of libtorch was built, since it likely used different dependencies than the ones we use, so would like to see the full configuration for that. We'll then need additional information to try to reproduce the problem, so please send the full command lines for all of the commands you used to set up and build with as well.

Thank you!

#3 - 03/15/2021 12:44 PM - Alex Wilkinson

Hi Erica, sure, so to setup I've been doing:

```
cd $WORKSPACE
curl -o libtorch-shared-with-deps-1.6.0%2Bcpu.zip https://download.pytorch.org/libtorch/cpu/libtorch-shared-wi
th-deps-1.6.0%2Bcpu.zip
unzip libtorch-shared-with-deps-1.6.0%2Bcpu.zip
mkdir test
cd test
<put attached CMakeLists.txt and test_tscript.cpp here>
mkdir build
mkdir buil_lar
```

Then to build the downloaded version:

```
cd $WORKSPACE
source /cvmfs/dune.opensciencegrid.org/products/dune/setup_dune.sh
setup gcc v8_2_0
setup cmake v3_9_0
cd test/build
cmake -DCMAKE_PREFIX_PATH=$WORKSPACE/libtorch_1-6-0 ..
cmake --build . --config Release
```

To build with the libtorch on cvmfs I start with a new shell and do:

```
cd $WORKSPACE
source /cvmfs/dune.opensciencegrid.org/products/dune/setup_dune.sh
setup gcc v8_2_0
setup cmake v3_9_0
setup libtorch v1_6_0 -q e19
cd test/build_lar
cmake -DCMAKE_PREFIX_PATH=/cvmfs/larsoft.opensciencegrid.org/products/libtorch/v1_6_0/Linux64bit+3.10-2.17-e19
 ..
cmake --build . --config Release
```

I've been building on dunebuild01 and running the test script on the dunegvpms.

I'm not sure how to give you the configuration of the downloaded libtorch, I'm using it untouched from the PyTorch download. The dependencies will definitely be different and I imagine that's responsible for the difference in timings. Maybe the memory usage is a feature of this rather than a genuine performance problem... in which case I will grit my teeth and make smaller networks!

Thanks, Alex

#4 - 03/18/2021 02:56 AM - Andrew Chappell

Hi Alex,

For your reference, the details of the CVMFS build of libtorch are, I believe (Patrick may be best placed to correct me if I'm wrong), here:
https://cdcvs.fnal.gov/redmine/projects/build-framework/repository/libtorch-ssi-build/revisions/issue-24669/entry/build_libtorch.sh

This was established as part of this rather long thread (https://cdcvs.fnal.gov/redmine/issues/24669) addressing performance issues (starting with LibTorch 1.5) while still accommodating the requirements of running in a batch environment.

The compile options starting around line 375 are probably most relevant in terms of the libtorch build itself and, as you note, it's unlikely that the version you link to above was built with similar flags.

#5 - 03/22/2021 10:34 AM - Kyle Knoepfel

A modified build of libtorch 1.6.0 has been available via the UPS product libtorch v1_6_0a, also installed in LArSoft's CVMFS area. Could you please check if you observe the same memory issues with that build?

```
setup libtorch v1_6_0a -q e19
```

#6 - 03/23/2021 06:36 AM - Alex Wilkinson

Hi Kyle,

The memory issue persists with 1.6.0a - the memory usage and timings are nearly identical to 1.6.0.

#7 - 03/23/2021 06:52 AM - Alex Wilkinson

I have built libtorch from source to better understand the configuration. To do this I ran the following commands:

```
cd $WORKSPACE
source /cvmfs/dune.opensciencegrid.org/products/dune/setup_dune.sh
setup gcc v8_2_0
setup cmake v3_9_0
setup python v3_7_2
python -m venv .
source bin/activate
pip install pyyaml

git clone https://github.com/pytorch/pytorch.git
cd pytorch
git checkout tags/v1.6.0
git submodule update --init --recursive
mkdir pytorch-build
cmake -DBUILD_SHARED_LIBS:BOOL=ON -DCMAKE_BUILD_TYPE:STRING=Release -DPYTHON_EXECUTABLE:PATH=`which python3` -
DCMAKE_INSTALL_PREFIX:PATH=../pytorch-install ..
cmake --build . --target install
<wait for hours>
```

Then to run my test script I changed the cmake line from my earlier post to

```
cmake -DCMAKE_PREFIX_PATH=$WORKSPACE/pytorch/pytorch-install ..
```

The memory usage and timings of this build are in line with original libtorch download (Maximum resident set size (kbytes): 2282840 and Elapsed (wall clock) time (h:mm:ss or m:ss): 0:28.12). I believe the configuration can be found in https://github.com/pytorch/pytorch/blob/v1.6.0/CMakeLists.txt at line 115.

#8 - 03/29/2021 10:37 AM - Kyle Knoepfel

Alex, Andy's comment above is relevant. Based on the information you have provided thus far, it appears that the libtorch version you are building is taking advantage of hardware-specific details under the covers (using system defaults, etc.). This is not conducive to being able to run on many different kinds of machines as you would find on the grid.

The build of libtorch provided by SciSoft can be used on any supported grid nodes, but that constrains how we build the libtorch product (in this case, using openblas as the BLAS backend, which has runtime selection of the hardware micro-architecture). So although you are likely to have larger memory usage than if you built and installed the product on your own machine, the processing performance should be comparable. If you find this is not the case, please let us know.

#9 - 04/12/2021 10:34 AM - Kyle Knoepfel

- % Done changed from 0 to 100

- Status changed from Feedback to Resolved

Please let us know if you run into any other issues.

Files

test_tscript.cpp	1.54 KB	03/09/2021	Alex Wilkinson
CMakeLists.txt	262 Bytes	03/09/2021	Alex Wilkinson